On Search Friction of Route Discovery in Offchain Networks

Saar Tochner[1] Stefan Schmid[2]
[1]The Hebrew University of Jerusalem [2]Faculty of Computer Science, University of Vienna

***Abstract*—Offchain networks provide a promising solution to overcome the scalability challenges of cryptocurrencies. However, design tradeoffs of offchain networks are still not well-understood today. In particular, offchain networks typically rely on fees-based incentives and hence require mechanisms for the efficient discovery of "good routes": routes with low fees (cost efficiency) and a high success rate of the transaction routing (effectiveness). Furthermore the route discovery should be confidential (privacy), and e.g., not reveal information about who transacts with whom or about the transaction value. This paper provides an analysis of the "search friction" of route discovery, i.e., the costs and tradeoffs of route discovery in large-scale offchain networks in which nodes behave strategically. As a case study, we consider the Lightning network and the route discovery service provided by the trampoline nodes, evaluating the tradeoff in different scenarios also empirically. Finally, we initiate the discussion of alternative charging schemes for offchain networks.**

I. Introduction

Despite the high popularity of cryptocurrencies, it remains a challenge to make fast payments at scale. This is mainly due to the inefficiency of the underlying consensus protocol: it can take several minutes until a transaction went through a full consensus and can be confirmed. A promising solution are emerging payment channel networks such as the Lightning network, which allow to perform transactions off-chain and in a peer-to-peer fashion: without requiring consensus on the blockchain. In a nutshell, a payment channel is a cryptocurrency transaction which escrows or dedicates money on the blockchain for exchange with a given user and duration. Users can also interact if they do not share a direct payment channel: they can route transactions through *intermediaries*.

However, the design of secure and scalable offchain networks is challenging and still not well-understood. In particular, these networks must not only be scalable but also account for strategic (i.e., selfish) user behavior; it must further be ensured that these networks do not introduce new security issues. A common approach to incentivize network nodes (the intermediaries) to contribute to the transaction routing is to use a fee-based mechanism: intermediaries can charge nodes which route through them a nominal fee. This is also the approach taken in the Lightning network which serves us as a case study in this paper.

This raises the question of how nodes can *discover* routes through intermediaries. One aspect here is cost efficiency: since different routes come at different fees, nodes require scalable mechanisms to find "short" (i.e., low-cost) routes. However, routes do not only have to be cheap but also provide sufficient liquidity to route the transaction: the route discovery mechanism should ensure a high success rate of the transaction routing; this property is known as effectiveness in the literature. Effectiveness is not only a performance concern: a lengthy discovery process may also jeopardize privacy, potentially leaking information about who aims to transact (i.e., find a route) with whom. Last but not least, the route discovery should be incentive-compatible, e.g., account that nodes are only willing to distribute routes from which they can benefit (e.g., which go through themselves).

Providing an effective and scalable route discovery is particularly challenging as large-scale off-chain networks are expected to be highly dynamic, e.g., due to the frequent changes of channels and fees. This renders solutions requiring wallets to keep up-to-date state information about the networks impractical. An interesting recent solution to reduce the burden on wallets, is the deployment of route discovery servers, such as the trampoline nodes in Lightning: these servers (which have more resources) maintain routes so that a wallet just needs to know how to reach the route server nodes in its neighborhood and can then request the desired route.

This paper provides an analysis of the efficiency-privacy tradeoff of off-chain route discovery, considering the Lightning network as a case study. In particular, we investigate to which extent route discovery can be efficient and effective, incentive compatible and confidential. Here, confidentiality is about more than just the actual data that is communicated in the discovery process, e.g., the source, the destination, or the transaction size; it is also about the possible metadata that is communicated implicitly, e.g., about the rate or time at which transactions occur. In fact, existing cryptographic techniques such as private search on key-value stores [7] can be used to provide data confidentiality, however, as we will show in this paper, nodes may still leak information about the frequency of transactions, i.e., about their *transaction rate*, to other nodes which are not on the transaction route.

We quantify the "search friction", i.e., the cost of the route discovery process, both analytically, deriving cost lower bounds, as well as empirically, exploring evaluations of these tradeoffs in real payment channel networks. Our results motivate research into alternative economic models to provide routing incentives which come at lower search friction costs, which we also start to discuss in this paper.

The remainder of this paper is organized as follows, see also Figure 1. We introduce a model for route discovery and provide a formal analysis in Section II. We report on our

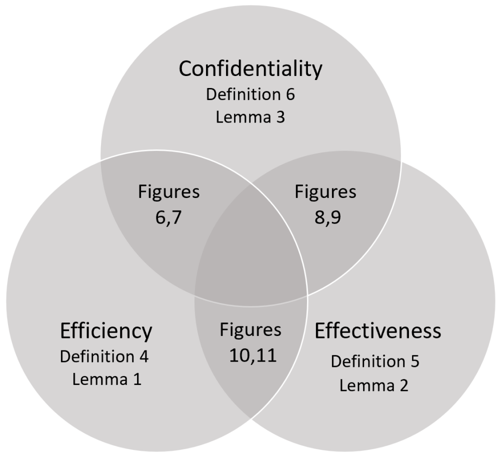

Fig. 1. Paper organization

empirical results in Section III. After reviewing related work in Section IV, we conclude our contribution and discuss future research directions in Section V.

II. Tradeoff Analysis of Route Discovery

In order to analyze the search friction and tradeoffs of route discovery, we consider the following simplified model. We model the offchain network as a graph of *channels*: two nodes can create (and may later delete) a payment channel between each other, to perform offchain transactions. This channel may also be used by other nodes for routing their transactions indirectly, i.e., using multi-hop routing. In this paper, we are primarily interested in two properties of a channel: the fee other nodes are charged to use this channel, and the channel capacity (resp. liquidity), which determines the size of the transactions it can support. This is a simplification of actual systems such as Lightning, but captures their essence.[1]

Definition 1 (Offchain Network)**.** *The offchain network is a weighted, directed graph $G(V, E)$, where V are the nodes in the network and E are the channels. A channel in the network $e \in E$ is characterized by a weight $w_e \in \mathbb{R}_+$ and capacity $w_{cap} \in \mathbb{R}_+$.*

The weight w_e is kept general here but typically is a function of the fee that the source node pays for an intermediate channel, the channels' age (older channels may be assumed to be more reliable), or previous knowledge (e.g., channels which failed in the past), among other.

The network is dynamic, i.e., channels may be added and removed over time, fees updated, etc., and hence nodes require mechanisms to discover the topology and learn about updates to be able to route their transactions through the offchain network, either explicitly or implicitly. For example, Lightning includes gossiping and active probing mechanisms to allow nodes to learn about routing fees. It is however more challenging to learn about the capacity w_{cap}; such information is typically not distributed for privacy reasons and hence, finding a path with sufficient capacity may require trial and error [20].

[1]E.g., nodes can set min/max values for channels, create non-public channels which cannot be used for multihop routing, etc.

More formally, a route on a topology $G(V, E)$ from $s \in V$ to $t \in V$ is a list of channels $e_1 \cdot \cdots, e_n \in E$ such that the source node of e_1 is s, and the target node of e_n is t and for every i, the target of e_i is the source of e_{i+1}. A transaction in the network is a payment from a source to a target along a "valid route": A route can serve a transaction of size l if every channel e along the route has enough capacity, i.e. $e_{cap} \geq l$. We assume that the weight of a route is simply the sum of the weights of its channels.

Inspired by existing offchain networks such as Lightning, in the following definition we will distinguish between types of nodes in the network

Definition 2 (Wallets and Trampoline Nodes)**.** *There are two types of nodes in the network: Wallets are simple nodes that do not have the resources to store and maintain information about the entire network. Trampoline nodes are nodes with more resources, that know the current network and provide information upon request.*

We are interested in exactly this discovery process, where wallets rely on the interaction with one or multiple trampoline nodes to find routes for their later transactions.

This route discovery process however introduces the following challenges:

- *Strategic behavior and efficiency:* Trampoline nodes may act selfishly and may only have an incentive to share routes which include themselves, such that they can charge the fee. As a consequence, a wallet may not learn about the most efficient (i.e., lowest cost) route. We assume that the TNs are honest (i.e. participate in the protocol) and can only manipulate their responses to routing queries.
- *Effectiveness:* Also related to the above, wallets may have to invest more resources into the discovery of efficient routes, exploring additional alternative trampoline nodes. The effectiveness of this route discovery process is further affected by the fact that not all the discovered routes may provide sufficient liquidity (i.e., capacity) for a large transaction which needs to be routed.
- *Privacy:* Through the repeated interactions with multiple trampoline nodes, querying for specific routes, a wallet may reveal confidential information about its transactions.

We are interested in the following family of route discovery algorithms:

Definition 3 (Routing Discovery Algorithm (RDA))**.** *A q-route discovery algorithm (q-RDA) is an algorithm that given a pair $s, t \in V$, performs at most q queries, issued to q trampoline nodes, and either returns a valid route or decides that this is not possible.*

We measure the quality, i.e., the efficiency, of a route found by the RDA, by comparing it to the *optimal route* with respect to the weight function on the topology.

Definition 4 (Efficiency)**.** *The efficiency of a route R from s to t is defined by the stretch, i.e., $\frac{w(R_{src,dst})}{w(O_{src,dst})}$, where $w(\cdot)$ is the weight of the route, and $O_{src,dst}$ is the route with the minimal weight between s and t. The weight of a route is the sum of its link weights.*

As discussed above, some routes in off-chain networks may be temporarily unavailable (e.g., due to offline nodes or lack of liquidity) and thus invalidate the result of the RDA. Another important metric to evaluate RDA hence concerns the number of queries it needs to issue until a valid route is discovered. For example, in the Lightning network, an available route is searched as part of the route initialization procedure. This process locks the channels along the route (a designated amount) and the channel commits to participate in the transaction.

Definition 5 (Effectiveness)**.** *The effectiveness of an RDA is the number of queries which have to be issued to successfully execute a given transaction.*

Furthermore, as transactions are privacy critical, the RDA should not leak any confidential information. Naturally, a first concern regards the information provided by the query directly, including e.g., source and destination nodes, potentially the transaction size, the resulting routes, etc. As discussed above, today we understand fairly well how to protect such information, e.g., using homomorphic encryption schemes and private information retrieval [13], [9], [6], [7]. However, there is another concern, related to the meta-data revealed from the query, e.g., the timestamp or even the existence of the route discovery query itself. While there also exist solutions to metadata private messaging systems, e.g., [11], [12], [19], we will show in the following that there is an inherent limitation what can be achieved in terms of an efficient and confidential route discovery. To this end, we introduce the notion of *leak rate*: to what extent can a node learn about the number of transactions in the network in a given time unit? That is, the leak rate is defined as the number of transactions in a single time unit that a node can learn about for a given set of transactions T under a given route discovery algorithm A.

Definition 6 (Leak Rate)**.** *An RDA A leaks at rate k if in order to route a transaction, k times more nodes will learn about the existence of this transaction compared to a scenario where the transaction is simply routed along the shortest path (e.g., using source routing).*

To clarify and motivate this notion, we give an example in Figure 2. In this simple network, a node learns about a transaction it should in principle have no idea about.

Efficiency. With these concepts in mind, we now first analyze the efficiency achievable by general route discovery algorithms. The following lemma shows an inherent cost of the route discovery process in the off-chain model

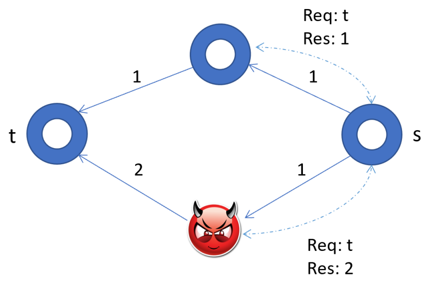

Fig. 2. Nodes can learn about a transaction although the request itself does not hold any information. In this simple topology, the red node learns that s performed a transaction with t just because of the query itself.

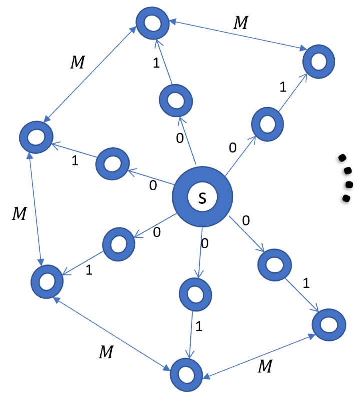

Fig. 3. Example with high cost: If there are $q+1$ direct neighbors, then any q-RDA struggles to find an efficient route.

Lemma 1. *For every q-RDA and every $M \in \mathbb{R}_{\geq 1}$ there exists a topology in which an RDA will return a route with weight M times higher than the optimal route, or it will not return a route at all.*

Proof. Consider the topology in Figure 3. Given a q-RDA A, we build a topology in which A will return a route with weight larger than M although there exists a route with weight 1. In our topology, the source node, s, is in the center, and is connected to $q+1$ TNs with channel weight 0. Each TN is connected to one unique node with a channel of weight 1, and all these nodes form a clique (i.e., are connected to one another) by channels of weight M. We will choose the target from one of these nodes. The RDA A queries q TNs in an order that is independent on target node (because s does not know the topology). But there are $q+1$ possibilities to the target, therefore there exists a node in the outer circle, t, that A does not query its direct TN neighbor. Choose this t to be our target. As the TNs are selfish, they will tell A only about routes that go through themselves, and none of them is directly connected to t; thus all the weights that A sees are at least of size $M+1$. Finally, A will return either a route with weight $M+1$ or no route at all, although the actual shortest route is of weight 1. □

In general, the efficiency will depend on the specific topology. To give an example, consider the complete network.

Example 1. *In a clique where all the weights are equal, the efficiency, in terms of the stretch, is upper bounded by 2.*

To see this, assume r is the weight of each channel. The optimal route is the direct channel, which weighs r. On the other hand, the route to each TN and the route from the TN to each target are also of weight r (in a clique there exists a direct channel), which gives $2r$ in total.

Effectiveness. We next consider the effectiveness a route discovery algorithm can achieve. Also here, we first derive a negative result for the general scenario.

Lemma 2. *For every q-RDA and every $M \in \mathbb{N}$, there exists a topology in which the first M routes from the algorithm will be unsuccessful.*

Proof. As in Lemma 1, we will build a topology in which the effectiveness of the RDA A will be as needed. The topology is composed from $M+2$ nodes: M nodes form a clique, one of the nodes in the clique is a TN, to which the source is connected, and the target is connected to the other nodes in the clique. There are $\sum_{k=0}^{M-2}\binom{M-2}{k}k!$ unique routes in the clique. A is not aware to the availability to the channels in the topology, therefore it offers the routes in an order which is independent to it. Now, define the channels to be unavailable in any of the first M routes. Therefore the first M routes that A will offer will be unavailable. □

Again, more specific networks can provide better guarantees.

Example 2. *In a scale-free network with n nodes, where all the channels are bi-directional, and where p is the probability that a channel has sufficient capacity, an q-RDA that queries the highest-degree nodes succeeds with a probability of at least $1-(1-p^{\frac{2\cdot log(n)}{\log\log(n)}})^{n\cdot(1-2^{-q})}$. For example, if a channel accepts a route independently with probability $p = 0.2$, for a network of size $n = 4000$ (e.g., Lightning), when the RDA queries only $q = 5$ TNs, then the probability to get at least one effective route is ≥ 0.999.*

To see this, note that the diameter of this network is $\frac{\log(n)}{\log\log(n)}$ on average (following [4]), therefore the length of the path from every source to every TN, and then from the TN to the target is bounded by $2\cdot\frac{\log(n)}{\log\log(n)}$. Therefore, the probability for each TN's suggestion to route the transaction is at least $p^{2\cdot\frac{\log(n)}{\log\log(n)}}$. Moreover, the number of paths from TN to a target is at least its degree, and the total degrees of the q nodes with the largest degree is $n\cdot(\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^q}) = n\cdot(1-2^{-q})$.

Confidentially. To which extent can we avoid rate leakage of the route discovery algorithm? The following lemma shows that if nodes behave strategically, we cannot upper bound the leaking rate of a route discovery algorithm.

Lemma 3. *For every q-RDA, and every $M \in \mathbb{R}$ there exists a topology in which the algorithm leaks at rate $\min\{M, q\}$.*

Proof. The proof is similar to the proof of Lemma 1, however, rather than worrying about the weight of the discovered route, we only worry about the existence of a valid route. Therefore, we will define our topology with a set of disconnected nodes, among the nodes in the outer circle (instead of clique). More specifically, our topology is simply a star topology, where the source is in the center, its neighbors are M TNs, and their neighbors are the possible targets (and they are all sinks, i.e. without outgoing channels). Clearly, if the RDA A will query a TN which is not the direct neighbor of the target, then it will return nothing (because there is no route). Let us choose our target to be the node that is connected to the M'th TN that A queries. If $M < q$ then the A will query M TNs until it will find a valid route; otherwise it will stop after querying q nodes unsuccessfully. In either way, A queries $\min\{M, q\}$ TNs. □

Example 3. *In scale free networks, the number of queries that we need to perform in order to find a "good" route is small.*

For example, in terms of betweenness, in a scale free network the betweeness centrality is distributed according to a power-law and relatively to the node's degree [3]. Therefore if the highest degree nodes are TN, and we ask the top k, then the TN will be on the optimal route to the target with high probability. Therefore the number of queries that we need to do in order to find an optimal route is exponentially small compared to the number of nodes in the topology.

III. Empirical Evaluation

In order to complement our theoretical results and in order to study the efficiency, effectiveness and confidentiality tradeoff in real networks, we consider the Lightning network as a case study for our experiments in the following.

A. Methodology

Following the Lightning network RFC regarding trampoline nodes[2], we assume that a wallet node stores its local knowledge on close trampoline nodes (TN) in the network, and can search for them within a close neighborhood. The wallet then queries some of the TNs for a route to the desired target, and will finally use the route with the lowest weight.

Collected data. We collected data about the Lightning network using a live Lightning node (lnd) which is connected to the mainnet through bitcoind. To extract the network structure (currently, the whole topology is stored by all nodes) we use the command *lncli describegraph*. The data used for this paper was retrieved on March 24th 2020, and provides information about the public channels.

Topologies. We consider two topologies in our experiments: The first is the Lightning network snapshot (which was studied e.g. in [16]), and the second is a synthetic sparse topology that was built using the following algorithm: create 1000 nodes in a circle, add one outgoing channel from each node to the node next to it, and another channel to other random node (total of 2 outgoing channels). The fee is the

[2]see *https://github.com/lightningnetwork/lightning-rfc/pull/654*

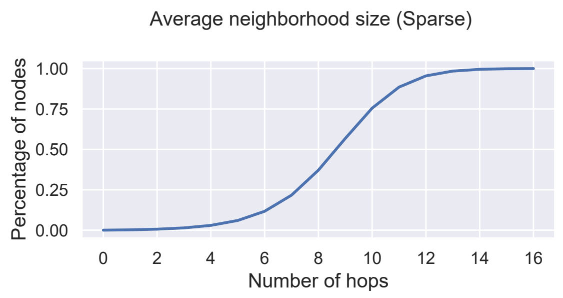

Fig. 4. Percentage of nodes as a function of the neighborhood size in the sparse topology.

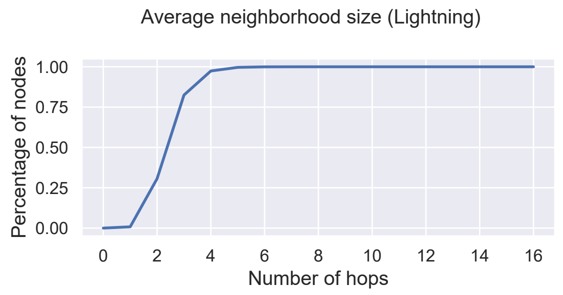

Fig. 5. Percentage of nodes as a function of the neighborhood size in the Lightning network's topology.

same for all the channels and equals 1. We choose this as the generative method because we want to follow Lightning's average out-degree while maintaining a strongly connected graph. Figures 4 and 5 show the number of nodes as a function of the neighborhood size in the two topologies. This comparison between the two topologies will later explain the key differences in the evaluations.

Graph weights. The weights of the edges in the Lightning network's graph are determined by the transaction size. Each channel determines base and proportional fee, and the final fee is $base\_fee + transaction\_size \times proportional\_fee$. Moreover, the fees are calculated backward from the target to the source, because the nodes should pay the fee for transferring the fees to the later intermediate nodes. We decided to neglect this backward computation due to the fact that practically it does not change the routes (as our additional experiments show). We further determined the transactions' size to be 10^6 milisatoshis. We finally determine the weight of each channel to be $base\_fee + 10^6 \times proportional\_fee$.

Transactions distribution. In the following, we assume that transactions follow two possible distributions: (i) one where all pairs of nodes in the network attempt to create a transaction and (ii) one where there are nodes with higher probability to execute a transaction (higher "activity level"). In the latter case, we determine a power-law distribution and grant it uniformly to the nodes. In particular, we uniformly partition the nodes to groups of size 100, and give the i'th group an activity level of 2^{-i}. Note that we need to model the transactions because transactions in the Lightning network are private by design. It is hard to infer the real distribution of the transactions since (i) information about transactions is hidden in the private state of channels and since (ii) routes are obscured by onion encryption.

Implementation details. We use the Floyd–Warshall all pairs shortest path algorithm to compute the optimal routes. Moreover, in order to keep shortest paths by limiting the number of neighborhood sizes (for example to search for the weight to the trampoline nodes in a neighborhood with a certain size), we use the "min-plus matrix multiplication" (or "distance product") algorithm, and stored the weights matrices for each number of hops. Finally, to find all the paths between specific source to target, we use the python module $networkx$.

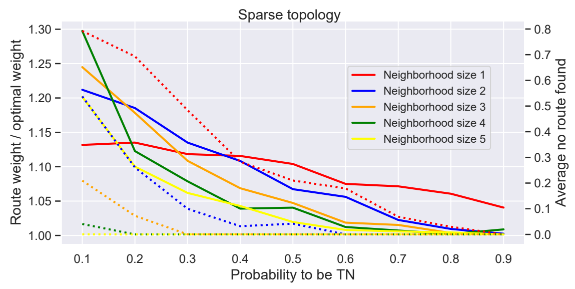

Fig. 6. Probability of not finding a route.

B. Tradeoff Evaluation

Efficiency-confidentiality tradeoff. We first evaluate the efficiency of the routes, depending on the neighborhoods in which trampoline nodes are searched. We already know from Lemma 1 that the efficiency can be low in the worst case, and we are now interested the efficiency in our specific examined topologies. The efficiency-confidentiality tradeoff can help a wallet to decide on the neighborhood size that it should query in order to find an efficient route.

We first consider our synthetic sparse topology: Figure 6 shows that the efficiency in small neighborhoods is better only because we cannot find many routes in this scenario. In the sparse topology this makes sense: there are less edges, therefore if there is a TN in a close neighborhood, then the optimal route goes through it with higher probability. For the Lightning network topology, in Figure 7, we can see that nodes find more TNs in their close neighborhoods, but the weights are far from optimal. One possible reason is that the topology is scale-free, therefore close neighborhoods are crowded, and the routes and the TN that the node finds are not on the optimal route (thus the resulting routes have "detours").

Effectiveness-confidentiality tradeoff. We next examine the RDA's effectiveness, i.e., the case in which the routes may be unavailable. As we showed in Lemma 2, highly ineffective queries may result when TNs offer unavailable routes, in which case the wallet will have to query many TNs and lose confidentiality. The next experiment explores this tradeoff, parametrized by the neighborhood size from which the wallet

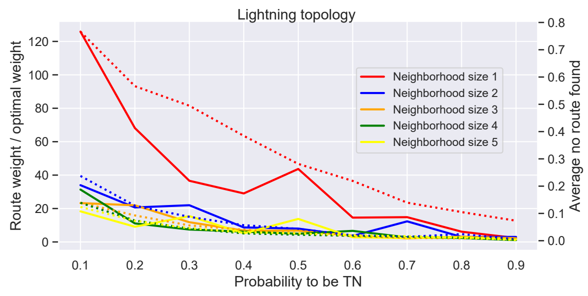

Fig. 7. Probability of not finding a route.

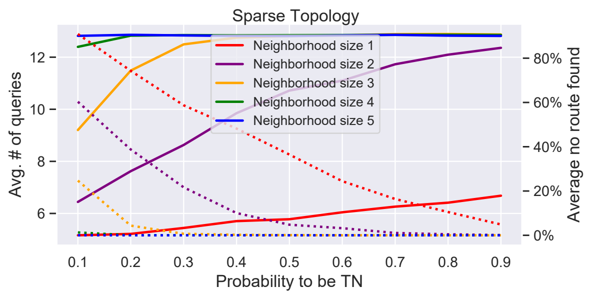

Fig. 8. Privacy-effectiveness tradeoff.

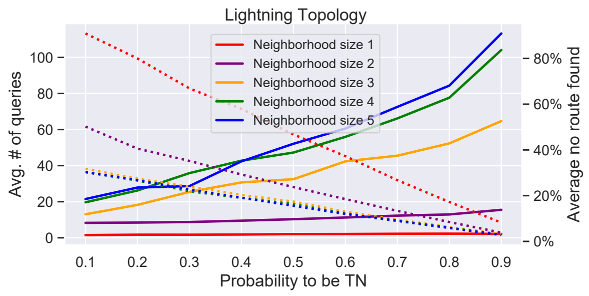

Fig. 9. Privacy-effectiveness tradeoff.

chooses the TN, and by the percentage of TNs in the network. We expect to see that as there are more TNs, the number of unsuccessful routes decreases (we will be offered more routes) and the number of queries will increase. For each pair of nodes in the topology, we ordered randomly all the TNs in the relevant neighborhood, and queried them one by one, each for 5 routes. Only if all the 5 routes failed, then we query another one.

As we discussed before, the transactions in the Lightning network are private, both in terms of the participating parties and the transaction size. In our experiment we hence simulate the unavailability of the nodes by considering only the lack of liquidity. For each channel, we simulate the occupied liquidity using a random variable $v \sim Uniform[0,1]$, and define the already-locked liquidity to be $v \cdot tx_{size} \cdot factor$; here tx_{size} is the transaction size that we try to route, which we set to 10^6 millisatoshis. The x-axis in the graphs indicate this factor.

In Figure 8, we fix the probability at which a channel will accept the transaction to 60%, and examine the number of queries that the wallet will have to execute before finding a route that accepts the transaction. We see that when we query bigger neighborhoods, the number of queries increases due to longer routes (which decreases the availability), but also more routes are found because there are more TNs to query (and thus more *disjoint routes*, i.e. different channels).

Figure 9 shows this tradeoff on the Lightning network. As in the sparse network, we see that larger neighborhoods will result in more available routes, but at the cost of more queries. Note the curvature of the graph, compared to the sparse topology, which shows that the number of queries does not increase as fast when there are more TNs. We consider this, again, as a result of the scale-free properties of the Lightning network, and specifically the higher degrees; the 3-neigborhood contains almost the entire graph, thus the TNs will yield disjoint routes which increases the probability to availability.

Effectiveness-efficiency tradeoff. The last tradeoff discusses the average route weight when assuming that the routes could be unavailable. In this experiment, we tried to route through the 10 shortest paths, and note the average weight and the number of pairs that did not find any route.

Figure 10 shows the tradeoff between the fee and the effectiveness of the route. Unlike the previous section, where we stopped querying after we found a valid route, in here we continued to ask all the TNs, and deduce the percentage of routes that we cannot use, and the average fee of the one that we can. This figure also reveals an interesting phenomenon in which the fee decreases when the probability for availability decreases. This happens because the available routes become shorter, and thus the average fee decreases.

Figure 11 shows this tradeoff in the Lightning network. We follow the same methodology of the previous section, in which each channel accepts a transaction randomly, as a function of the transaction size, a uniformly generated number, and a factor. It is interesting to note that the last phenomenon that we described on the sparse topology does not hold here. The fee increases due to the larger number of pairs that succeed to transact. This might suggest that either there is a single route and if it is not available, then the transaction cannot be executed; or there are many different routes with approximately the same weight, which makes the unavailability of some routes have a smaller effect on the route efficiency.

C. Extensions

We next explore a generalization of trampoline nodes, which may further improve scalability. We observe that in principle, nodes can answer path request queries also if they just used the path. The intuition is that while nodes will not remember the entire topology they may still have a route cache. Let us hence consider “Partial Nodes” (PN), nodes that share their past knowledge using the same selfish mechanisms like TNs, when answering path request queries.

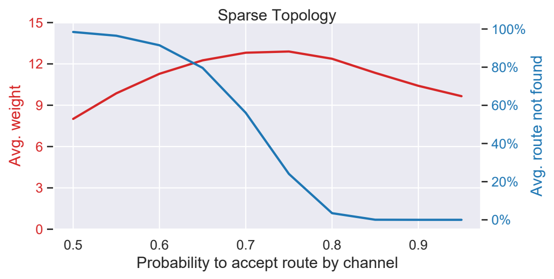

Fig. 10. Efficiency-effectiveness tradeoff in the sparse topology.

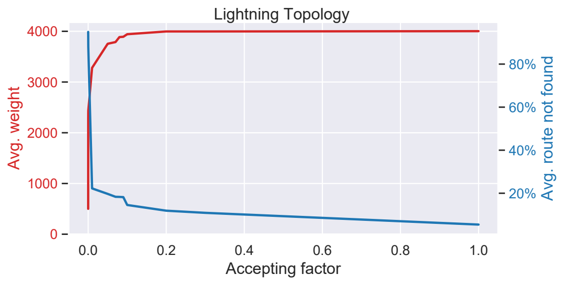

Fig. 11. Efficiency-effectiveness tradeoff in the Lightning network topology.

Note that in this extension, the wallet nodes will not benefit from a better effectiveness (just like the TNs, the PNs do not know the availability of the channels) or confidentiality (because they still query the same number of intermediate nodes). The major improvement that partial nodes will contribute to the network is the efficiency of the resulting routes. The are more nodes in the network that share the optimal routes to the target, therefore the "detour" of the route through the TN/PN will be smaller. Figure 12 assumes that each PN stores paths to 50 uniformly selected nodes. Here we fixed the number of TNs and the neighborhood size to obtain a clearer view of the improvement in the resulting routes. On the other hand, Figure 13 shows the limited benefits of "partial nodes" on the sparse topology. The effectiveness is low due to the small number of different routes to the target.

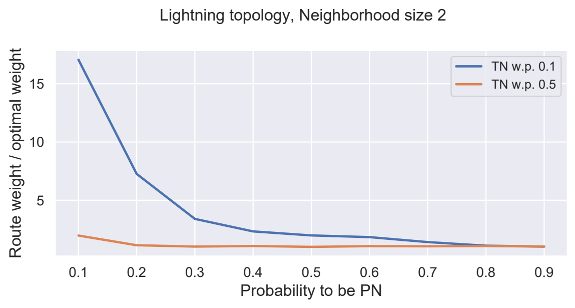

Fig. 12. Ratio between the resulting route and the optimal route using partial nodes (on the Lightning network topology).

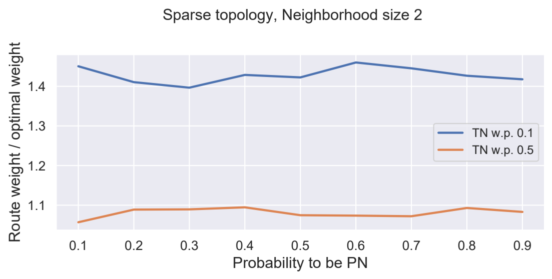

Fig. 13. Ratio between the resulting route and the optimal route with partial nodes (on the sparse topology).

IV. Related Work

A comprehensive discussion of the different aspects of route discovery in off-chain networks can be found in the SoK article by Gudgeon et al. [10], who also identify the challenges related to effectiveness, efficiency, scalability, cost-effectiveness and privacy. In general, over the last years, many clever route discovery algorithms have been proposed in the literature. Two well-known examples are SpeedyMurmurs [15] and SilentWhispers [8] which focus on the effectiveness and efficiency challenges; the authors for instance show empirically that SpeedyMurmurs can find routes by up to two orders of magnitude faster then the existing algorithms, while maintaining the same success ratio. However, these approaches face the challenge of cost-effectiveness in the presence of selfish behavior. Another interesting approach is pursued in the SpiderNetwork [17]: the payments are split into units and the route discovery algorithm routes each of them individually (similarly to a packet-switched network). This method however does not account for privacy and cost-effectiveness in selfish environments. In MAPPCN [18] the authors suggest and analyze a route discovery algorithm which preserves privacy but which does not account of the information leaking problem addressed in this paper.

From an economical perspective, our research is related to consumer-producer markets where producers offer products which are infinitely expansible (e.g., digital goods) [14], and where consumers may suffer from search friction: a cost in addition to the actual price [1]. One example is Pandora's problem [21] wherein Pandora needs to pay a fee to open boxes, and each box offers different value. In our context, the cost revolves around the loss of privacy and additional network overheads.

Last but not least, our analysis is in the spirit of classic models such as [1]: we model the interaction between the TN (which sells an item or service) and the wallets that consume it. Many different scenarios of this game were researched previously [2], [5], showing that pricing in the network will rise with the search friction.

V. Conclusion & Future Work

We modeled, analyzed and empirically evaluated the trade-off between efficiency, effectiveness and privacy of route discovery in offchain networks which come with scalability requirements and where node behave selfishly.

Our analysis and evaluation show that the tradeoffs depend on the underlying network topology. For example, while we have derived fairly negative results for worst-case topologies, in the Lightning network, the privacy-effectiveness tradeoff may be fairly acceptable, and also significantly better than for the sparse topology we considered. This is due to the many trampoline nodes in close neighborhoods, which yields many disjoint paths for only a few queries, but causes detours from the optimal routes. The effectiveness-efficiency tradeoff depends on the centrality of the network, where the efficiency drops fast with the effectiveness; e.g., if there is a single efficient route which fails, then either the efficiency drops significantly or we cannot find another route.

Our paper shows that route discovery service nodes, such as trampoline nodes, introduce an interesting new economic role in offchain networks. We believe that exploring their role, and their different incentives compared to the other network nodes, further, constitutes a very interesting avenue for future research. While the wallet has a search friction that is based on the number of queries that it needs to perform and the resulting privacy loss, the TNs may offer different prices that may be changed according to the wallet's strategy. The study of the resulting strategic behaviors and games may provide interesting insights, e.g., on whether overestimating the value of privacy will motivate the TN to increase the fee of the offered route.

Moreover, as discussed, the interaction between the TNs (the "sellers") and the wallets (the "buyers") is reminiscent of digital goods trading games. To the best of our knowledge, this paper is the first to establish this connection, considering the search-friction for the buyers. Investigating the differences between the traditional models (e.g., considering servers that offer files to download) and our setting (considering privacy and bandwidth) further, constitutes another interesting avenue for future research.

It could further be interesting to explore alternative strategic behaviors where, e.g., TNs aim to route transactions through worse routes in order to manage liquidity in the TN's channels, or where TNs do not follow the protocol in order to prevent information from their competitors. This also raises the question about efficient mechanism designs.

References

[1] Simon P Anderson and Regis Renault. Pricing, product diversity, and search costs: A bertrand-chamberlin-diamond model. *The RAND Journal of Economics*, pages 719–735, 1999.

[2] Pak Hung Au. Competition in designing pandora's boxes. *Available at SSRN 3141387*, 2018.

[3] Marc Barthelemy. Betweenness centrality in large complex networks. *The European physical journal B*, 38(2):163–168, 2004.

[4] Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24, 2004.

[5] Michael Choi, Anovia Yifan Dai, and Kyungmin Kim. Consumer search and price competition. *Econometrica*, 86(4):1257–1281, 2018.

[6] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[7] Hu et al. Private search on key-value stores with hierarchical indexes. In *2014 IEEE 30th International Conference on Data Engineering*, pages 628–639. IEEE, 2014.

[8] Malavolta et al. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *NDSS*, 2017.

[9] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[10] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Off the chain transactions. *IACR Cryptology ePrint Archive*, 2019.

[11] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *13th* $\{USENIX\} - \{OSDI\}$ *18*, pages 711–725, 2018.

[12] David Lazar and Nickolai Zeldovich. Alpenhorn: Bootstrapping secure communication without leaking metadata. In *12th* $\{USENIX\} - \{OSDI\}$ *16*, 2016.

[13] Eduardo Morais, Cees van Wijk, and Tommy Koens. Zero knowledge set membership. *none*, 2018.

[14] Danny Quah. Digital goods and the new economy. *none*, 2003.

[15] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748*, 2017.

[16] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. Topological analysis of bitcoin's lightning network. In *Mathematical Research for Blockchain Economy*, pages 1–12. Springer, 2020.

[17] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Mohammad Alizadeh, Giulia Fanti, and Pramod Viswanath. Routing cryptocurrency with the spider network. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 29–35, 2018.

[18] Somanath Tripathy and Susil Kumar Mohanty. Mappcn: Multi-hop anonymous and privacy-preserving payment channel network. *none*, 2020.

[19] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.

[20] Utz Nisslmueller, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. Toward active and passive confidentiality attacks on cryptocurrency off-chain networks. In *Proc. 6th International Conference on Information Systems Security and Privacy (ICISSP)*, 2020.

[21] Martin L Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, pages 641–654, 1979.